CS-Lab1-

Ahmed Alhasan, Yash Pawal, Mohsen Pirmoradiyan

1/26/2020

Question 1: Be careful when comparing

```
## [1] "Subtraction is wrong!"
```

```
## [1] "Subtraction is correct!"
```

A computer number is an exact value of a floating-point number. Given x as a real number $[x]_c$ is the floating-point number closest to x. So x is a computer number if and only if $x = [x]_c$. The computer numbers, therefor, do not correspond to the real numbers in a natural way. An integer is exactly represented by a computer fixed-point number, a real number, however, may or may not have an exact representation by a floating-point number. x as a real number to be represented by a computer number is rounded to the nearest floating-point number. An important point is that the computer numbers(fixed-pont and floating-point) are finite. Because the numbers are to be represented to a fixed number of bits. The fraction 1/3 in decimal form is actually 0.333… which is infinitely recurring, hence no exact representation for this real number exist by a computer floating-point. In fact the representation of 1/3 is a rounded number to the nearest floating-point. The fraction 1/12 also has the same situation as 1/3. It recurs infinitely and therefor it can not be accurately represented by a computer number. A rounding error, as a result, will exist in computations containing such fractions. The fractions 1/2 and 1/4 are finite numbers and can be represented accurately by a floating-point computer number. As a rsult of rounding error discussed above we get a wrong answer for the first comparison. Both sides of the equality are rounded to the nearest floating-point and these nearest floating points are not the same. By rounding these numbers so that they have a finite number of digits after decimal points, we will get a correct result:

```r
x1 <- 1/3
x2 <- 1/4
if (round((x1 - x2), 2) == round(1/12, 2)){
  print("Subtraction is correct!")
}else{
  print("Subtraction is wrong!" )
}
```

```
## [1] "Subtraction is correct!"
```

Question 2: Derivative

```
## [1] 1.110223
```

```
## [1] 0
```

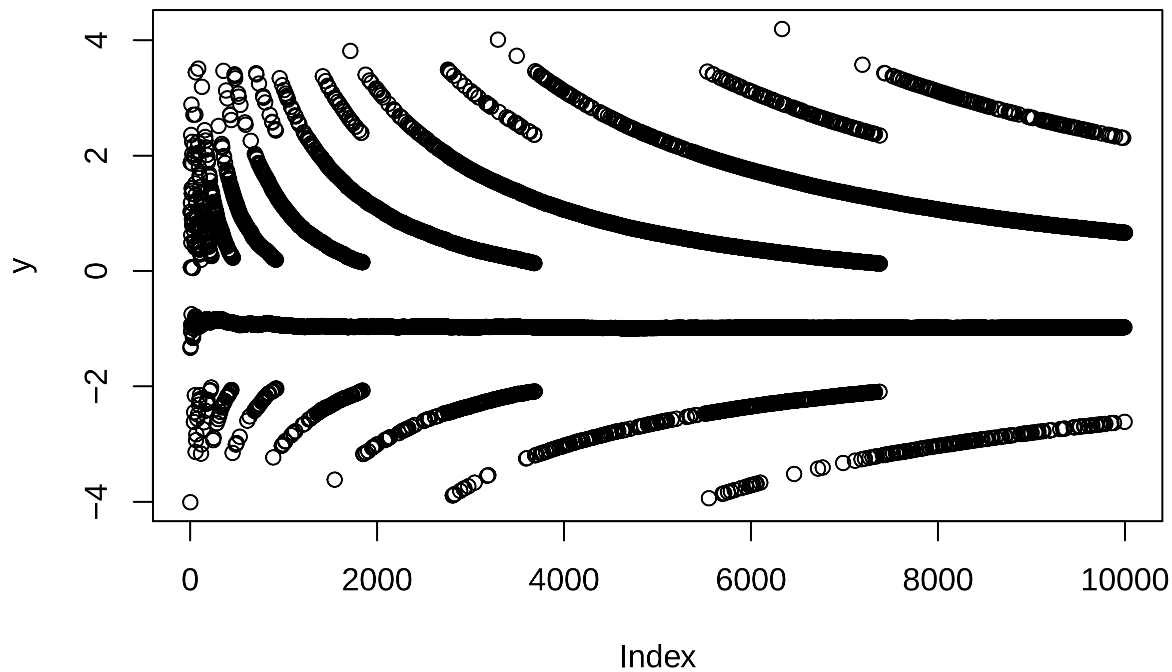
The true value for the derivative of $f(x) = x$ is 1. However, the result of this function for $x = 1$ is 1.110223 and for $x = 100000$ is 0. When $x = 100000$ it is very large compared to e and due to loss of significant figures the two quantities $((x + e)and(-x))$ are equal but with opposite sign values, so we get 0 as the result. In fact the very large value (100000) dominates the statements, so the significant digits of the very small number is lost. In other words loss of significant digits or underflow happens.

Discussion: There is a discussion within group members to name this phenomenon. We wonder if it can be referred to as "Cancelation" or not? One of the member believe in "canceletion" and so to avoid this error happening we should sort the numbers ascending: $10000 - 10000 + 1e - 15$ **so that the smaller number at the end of the terms will not be lost, however, this is not agreed by the other members.One of the other member belives that this happens just because of**

underflow and cancellation is not the case here and rearrangement should not be implemented since the function definition would be changed.

When $x = 1$

Question 3: Variance



Ideally the value of varianve should converge to 1, however for our function it doesn't due to catastrophic cancellation.Therefore, this algorithm may not be regarded as an acceptable approach for computing the variance. For the improvement we implement the cramer approach and we used the code provided on lisam by Krsysztof:

```
var_YC<-function(v_x){
## v_x is a numerical vector of length greater than 2
## this function calculates the sample variance
## using the Youngs and Cramer algorithm
    T <- v_x[1]
    RSS <- 0
    n <- length(v_x)
    for (j in 2:n){
    T <- T+v_x[j]
    RSS <- RSS+((j*v_x[j]-T)^2)/(j*(j-1))
    }
    RSS / (n-1)
}

var_YC(x)
```

```
## [1] 0.9762649
```

The result from this approach is reasonable. In the first algorithm very similar numbers cancel each other, however in this algorithm the subtractive cancellation is avoided.

Question 4: Linear Algebra

Not scaled data:

```
## Error in solve.default(A): system is computationally singular:
## reciprocal condition number = 7.13971e-17
```

The linear system does not have an answer as the matrix A is singular. This Matrix is not invertible. It can happen because of dependency between some variables,i.e, two or more variables are highly correlated. Ths will end in singularity in which the inverse of the matrix does not exist.

```
## The condition number:
##  1.157834e+15
```

The condition number is very high. If a matrix is singular then its condition number is very large.

For a well-behaved system $Ax = b$, a small change in b $(b + \delta b)$ will cause a relatively small change in $x(x + \delta x)$. It means that if δb is small we expect that the resulting solution $(\tilde{x})$ should be close to x. Such a system is well-conditioned, that is, if $\|\delta b\|/\|b\|$ is small, then $\|\delta x\|/\|x\|$ is likewise small. By definition:

$$\|\delta x\|/\|x\| \leq \|A\|\|A^{-1}\|\|\delta b\|/\|b\|$$

condition number with respect to inversion is $\|A\|\|A^{-1}\|$. As the condition number tends to infinity the upper bound of relative change in the solution caused by perturbation $\|\delta b\|/\|b\|$ increases. In other words the system is very sensitive to small changes and thus is very susceptible to roundoff error. We do not want this upper bound to be large, so a large condition number is bad.

In this question the condition number is very high and we may conclude that it is an ill-conditioned matrix.

Scaling the data set

```
##                    [,1]
## Channel1    -110.6123672
## Channel2    -221.2873564
## Channel3     378.1193651
## Channel4    -129.7293023
## Channel5     413.3177902
## Channel6     -79.6081556
## Channel7    -203.0804959
## Channel8      82.8265719
## Channel9    -132.4268940
## Channel10    255.8453173
## Channel11   -328.5537576
## Channel12   -304.2824757
## Channel13    624.2810079
## Channel14   -299.0199845
## Channel15     40.8283196
## Channel16   -257.6026907
## Channel17    169.2845086
## Channel18    296.6422779
## Channel19   -325.0603985
## Channel20     -3.0061504
## Channel21    554.5561922
```

```
## Channel22  -1366.0306884
## Channel23   1860.3712583
## Channel24  -1416.1508534
## Channel25    631.8507017
## Channel26   -112.0430143
## Channel27     17.0058292
## Channel28   -228.9169969
## Channel29    444.2652834
## Channel30   -597.3771973
## Channel31    438.1421237
## Channel32    315.0439168
## Channel33   -349.8128628
## Channel34   -285.9130097
## Channel35    418.5794391
## Channel36    -79.1066085
## Channel37   -305.9378992
## Channel38    284.2524830
## Channel39   -435.5696023
## Channel40    819.7566701
## Channel41   -885.0128709
## Channel42    324.5897799
## Channel43    524.5893652
## Channel44   -583.4383039
## Channel45   -140.1767449
## Channel46    577.2409424
## Channel47   -294.2702846
## Channel48    -68.0751871
## Channel49    -90.4927776
## Channel50    404.1462685
## Channel51   -699.0030347
## Channel52   1258.8888457
## Channel53  -1672.7374520
## Channel54   1486.2359579
## Channel55   -812.3647333
## Channel56    192.4958628
## Channel57    -32.9108742
## Channel58      7.3739491
## Channel59    -88.6896542
## Channel60    344.8764025
## Channel61   -454.3518890
## Channel62    447.6203573
## Channel63   -197.4180972
## Channel64    222.3366513
## Channel65   -399.2564804
## Channel66    364.8682783
## Channel67   -367.1635176
## Channel68    243.9238488
## Channel69    -76.2955745
## Channel70   -318.1918486
## Channel71    327.6656428
## Channel72   -178.5232382
## Channel73    119.1853879
## Channel74    445.1155355
## Channel75    -20.0131180
```

```
## Channel76   -642.7508884
## Channel77    369.4810726
## Channel78    -74.9013178
## Channel79    -23.4853654
## Channel80   -676.8615059
## Channel81   1013.4537410
## Channel82   -889.7622776
## Channel83    403.0065793
## Channel84    424.0848037
## Channel85   -801.0956082
## Channel86    655.0134198
## Channel87    659.1829737
## Channel88  -2150.8325565
## Channel89   1671.8088784
## Channel90    298.6977110
## Channel91   -332.1727810
## Channel92   -487.3689702
## Channel93    278.6277351
## Channel94    201.6627326
## Channel95   -609.5081418
## Channel96    565.2851754
## Channel97   -133.3407557
## Channel98   -368.0087287
## Channel99    238.2015991
## Channel100    24.6418181
## Fat           -1.6666403
## Moisture      -0.9341099
```

```
## The condition number:
##  490471520662
```

When we scale the data the round-off error becomes less significan, even though the new condition number is lower it is not necessarily well-conditioned, but we have lesser perturbation to deal with.